

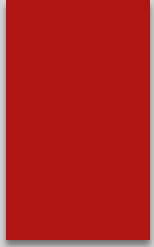


Session one

PHASE ONE



Our Plan



رب اشرح لي صدري
ويسر لي امري

Outline


- ▶ Data types
- ▶ Input/output
- ▶ Conditions
- ▶ Loops
- ▶ Functions

C Code !

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains C code for a program that prompts for a favorite number and prints a response.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(){
5     int favorite_number;
6     printf("Enter your favorite number:");
7     scanf("%d",&favorite_number);
8     printf("Amazing this is my favorite number too");
9
10    return 0;
11 }
```

C++ Code !



```
1 #include <iostream>
2 using namespace std;
3
4 int main(){
5     int favorite_number;
6     cout<<"Enter your favorite number from 1 to 100:";
7     cin>>favorite_number;
8     cout<<"Amazing these is my favorite number too";
9
10    return 0;
11 }
```

DATA TYPES

C++ primitive types

| Type Name | Bytes | Other Names | Range of Values |
|---------------|-------|-----------------------------|---|
| int | 4 | signed | −2,147,483,648 to 2,147,483,647 |
| bool | 1 | none | false or true |
| char | 1 | none | −128 to 127 by default |
| signed char | 1 | none | −128 to 127 |
| unsigned char | 1 | none | 0 to 255 |
| short | 2 | short int, signed short int | −32,768 to 32,767 |
| long | 4 | long int, signed long int | −2,147,483,648 to 2,147,483,647 |
| long long | 8 | none | −9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |
| float | 4 | none | 3.4E +/- 38 (7 digits) |
| double | 8 | none | 1.7E +/- 308 (15 digits) |
| wchar_t | 2 | __wchar_t | 0 to 65,535 |

- size is implementation-dependant
- table is from MSDN and refers to Microsoft Visual C++

VARIABLES

- ▶ not consistent or having a fixed pattern; liable to change

Rules for naming the variable

- All variable names must begin with a letter of the alphabet or an underscore(_)
- After the first initial letter, variable names can also contain letters and numbers.
- No spaces or special characters, however, are allowed. Uppercase characters are distinct from lowercase characters.
- You cannot use a C++ keyword (reserved word) as a variable name.

C++ keywords

Keywords in C++

Keywords in C++ A keyword is a reserved word. You cannot use it as a variable name, constant name etc. A list of 32 Keywords in C++ Language which are also available in C language

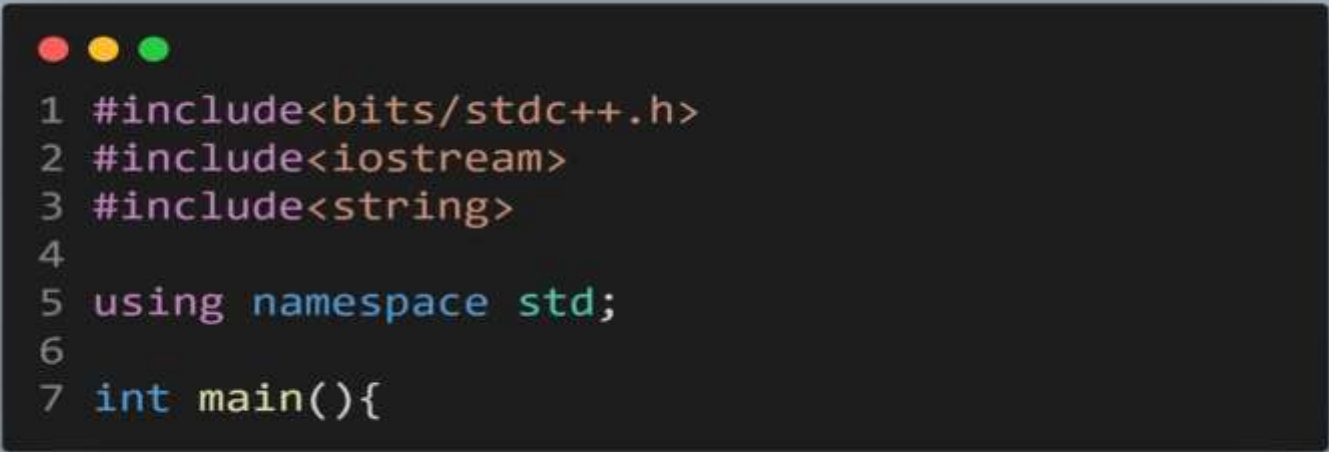
| | | | | | | | |
|--------|--------|----------|--------|----------|----------|----------|--------|
| auto | break | case | char | const | continue | default | do |
| double | else | enum | extern | float | for | goto | if |
| int | long | register | return | short | signed | sizeof | static |
| struct | switch | typedef | union | unsigned | void | volatile | while |

Declaration and initialization

```
1  #include<bits/stdc++.h>
2
3  using namespace std;
4
5  int main(){
6      //declaration
7      int number1 ;
8      float decimal ;
9      double decimal_2 ;
10     long long number_2 ;
11     bool on_off ;
12     char character ;
13     string word ;
14     //intialization
15     number1 = 2 ;
16     decimal = 12.6 ;
17     decimal_2 = 678957566.5676
18 ;
19     number_2 = 354676879655 ;
20     on_off = true ;
21     character = 'a' ;
22     word = "name";
23     int num1 , num2 , num3 = 4
24 ;
25     num1 = 3 ;
26     decimal = 6.098 ;
27 }
```

Preprocessor directives:

- ▶ Preprocessor directives are the type of macros and a phase before compilation takes place. It can be said that these are some set of instructions given to compiler to perform actual compilation.



```
1 #include<bits/stdc++.h>
2 #include<iostream>
3 #include<string>
4
5 using namespace std;
6
7 int main(){
```


The main function

- Every c++ have exactly one function
- Starting point program execution
- Return 0 indicate successful program runs
- The int indicate the return type of the function so it returns 0

using namespace std;


- Names given to parts of code to help to reduce naming conflicts
- Std is the name for c++ standard name space
- Scope resolution operator (::) :it is used to revolve which name we use.
- To make much easier to yourself you can write:
- (using namespace std;) instead of std:: before each identifier

without using namespace std;



```
1 #include <iostream>
2
3 int main(){
4     int favorite_number;
5     std::cout<<"Enter your favorite number from 1 to 100:";
6     std::cin>>favorite_number;
7     std:: cout<<"Amazing these is my favorite number too";
8
9     return 0;
10 }
```

by using namespace std;



```
1 #include <iostream>
2 using namespace std;
3
4 int main(){
5     int favorite_number;
6     cout<<"Enter your favorite number from 1 to 100:";
7     cin>>favorite_number;
8     cout<<"Amazing these is my favorite number too";
9
10    return 0;
11 }
```

C++ Comments

- ▶ Comments can be used to explain C++ code, and to make it more readable. It can also be used to prevent execution when testing alternative code.
- ▶ **Comments can be singled-lined or multi-lined.**
- ▶ `// This is a comment`
- ▶ `/* The code below will print the words Hello World!`
- ▶ `to the screen, and it is amazing */`

INPUT/OUTPUT

- ▶ `Cin :>>`
 - Standard input stream
 - Keyboard

- ▶ `Cout :<<`
 - Standard output stream
 - console/ / monitor

INPUT/OUTPUT

```
1  #include<bits/stdc++.h>
2
3  using namespace std;
4
5  int main(){
6      int x , z , y ,a;
7      cout<<"plz, Enter first number : ";
8      cin>>x;
9      cout<<"plz Enter a sacond & third number : "
10     ;
11     cin>>z>>y;
12     cout<<"number x is : "<<x<<"\n";
13     cout<<"number z is : "<<z<<"\t\t"<<"number y
14     is : "<<y<<endl;
15     a = z * x + y ;
16     cout<<"a = "<<a<<endl;
17 }
```

ARITHMETIC OPERATORS

Arithmetic Operators

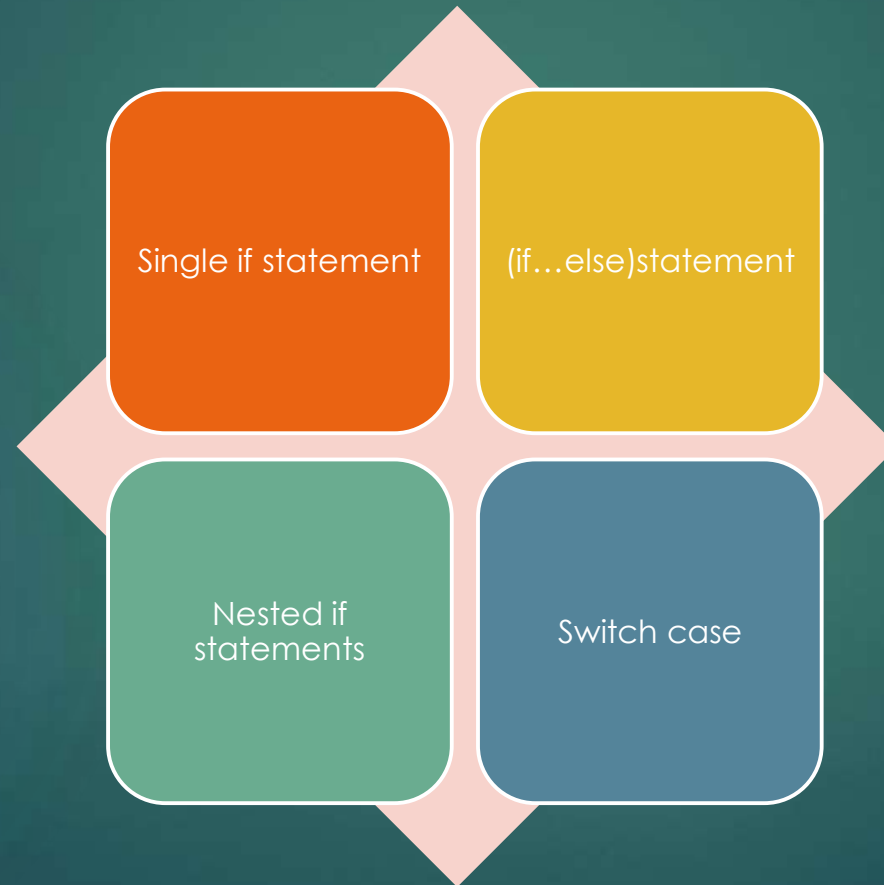
X = 20, Y = 40

| Operator | Description | Example |
|----------|----------------|---------------------|
| + | Addition | X + Y will give 60 |
| - | Subtraction | X - Y will give -20 |
| * | Multiplication | X * Y will give 800 |
| / | Division | Y / X will give 2 |
| % | Modulus | Y % X will give 0 |
| ++ | Increment | Y++ gives 41 |
| -- | Decrement | Y-- gives 39 |

problems

- ▶ Coins
- ▶ Problem - C

conditions



TO make any condition we must use comparison operators

| Operator | Use | Example |
|----------|--------------------------|-----------|
| < | Less than | if (a<b) |
| <= | Less than or equal to | if (a<=b) |
| > | Greater than | if (a>b) |
| >= | Greater than or equal to | if (a>=b) |
| == | Equal | if (a==b) |
| != | Not equal | if (a!=b) |

If condition

- ▶ **if (condition)**
- ▶ **{**
- ▶ **Statement;**
- ▶ **}**

```
1 #include<bits/stdc++.h>
2
3 using namespace std;
4
5 int main(){
6     int a = 10 , b = 9 ;
7     if (a>b){
8         cout<<"step 1"<<endl;
9     }
10    b++;
11    if(b<a){
12        cout<<"step 2"<<endl;
13    }
14    if(a>=b){
15        cout<<"step 3"<<endl;
16    }
17 }
```

(if...else) condition statements

- ▶ **if (condition)**
- ▶ **{**
- ▶ **Statement;**
- ▶ **}**
- ▶ **Else**
- ▶ **{**
- ▶ **statement;**
- ▶ **}**

```
1 #include<bits/stdc++.h>
2
3 using namespace std;
4
5 int main(){
6     int num ;
7     cout<<"plz, enter the number : ";
8     cin>>num;
9     if (num>0){
10         cout<<"The number is postive"<<endl;
11     }
12     else{
13         cout<<"The number is negative"<<endl;
14     }
15 }
```


Nested If

- ▶ **if (condition)**
- ▶ **{**
- ▶ **Statement;**
- ▶ **}**
- ▶ **Else if (condition)**
- ▶ **{**
- ▶ **Statement;**
- ▶ **}**
- ▶ **.....**
- ▶ **Else**
- ▶ **{**
- ▶ **statement;**
- ▶ **}**

```
1 #include<bits/stdc++.h>
2 #include<iostream>
3 using namespace std;
4
5 int main(){
6     long long x;
7     cin>>x;
8     if(x<0)
9         cout<<"Negative";
10    else if(x>0)
11        cout<<"Positive";
12    else
13        cout<<"Equals Zero";
14    return 0;
15 }
16
```

Switch case

- ▶ switch(expression) {
- ▶ case x:
- ▶ // code
- ▶ break;
- ▶ case y:
- ▶ // code
- ▶ break;
- ▶
- ▶ default:
- ▶ // code
- ▶ }

```
1 #include<bits/stdc++.h>
2 #include<iostream>
3 using namespace std;
4
5 int main(){
6     long long x;
7     cin>>x;
8     switch(x/10) {
9         case 9:
10             cout<<"excellent";
11             break;
12         case 8:
13             cout<<"very good";
14             break;
15         case 7:
16             cout<<"good";
17             break;
18         case 6: case 5:
19             cout<<"weak";
20             break;
21         case 4: case 3: case 2: case 1: case 0:
22             cout<<"fall";
23             break;
24         default:
25             cout<<"plz, Enter number 1 to 99 :";
26     }
27     return 0;
28 }
```

Conditional operator

► OR `||` AND `&&` NOT `!`
Suppose, `a = 5` `b = 8`

Then,

`(a > 3) && (b > 5)` evaluates to `true`
`(a > 3) && (b < 5)` evaluates to `false`

`(a > 3) || (b > 5)` evaluates to `true`
`(a > 3) || (b < 5)` evaluates to `true`
`(a < 3) || (b < 5)` evaluates to `false`

`!(a < 3)` evaluates to `true`
`!(a > 3)` evaluates to `false`

problems

- ▶ Problem – E
- ▶ Comparison

LOOPS

- ▶ **for loops**
- ▶ **while loops**
- ▶ **do .. while loops**

for loops

For (initialization ; condition ; update)

{

statement;

}

```
1 #include<bits/stdc++.h>
2 #include<iostream>
3 using namespace std;
4
5 int main(){
6     long long x,i;
7     cin>>x;
8     for ( i=1 ; i<=x ; i++){
9         if (i % 2 == 0)
10             cout<<i<<" is an even number."<<endl;
11         else
12             cout<<i<<" is an odd number."<<endl;
13     }
14     return 0;
15 }
```

While loop

- ▶ `while (/* condition */)`
- ▶ `{`
- ▶ `/* code */`
- ▶ `}`

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main(){
6     int x;
7     cout<<"Enter number:";
8     cin>>x;
9     while(x-->0)
10    {
11        cout<<"Hello world"<<endl;
12    }
13
14     return 0;
```

Do While

- ▶ **do**
- ▶ **{**
- ▶ **/* code */**
- ▶ **}**
- ▶ **while (/* condition */);**

```
1 #include<iostream>
2
3 using namespace std;
4
5 int main(){
6     int x;
7     cout<<"Enter a number:";
8     cin>>x;
9     do
10    {
11        cout<<"Hello world"<<endl;
12        x--;
13    }
14    while(x>0);
15    return 0;
16 }
```


Break continue

BREAK

A loop control structure that causes the loop to terminate and pass the program control to the next statement following the loop

Helps to terminate the execution of the loop

CONTINUE

A loop control structure that causes the loop to jump to the next iteration of the loop immediately

Helps to skip statements inside the loop

Visit www.PEDIAA.com

Break

continue

```
1 #include<iostream>
2
3 using namespace std;
4
5 int main(){
6     int n;
7     cin>>n;
8     for (int i=1;i<=n;i++)
9     {
10         if(i==2)
11             continue;
12         else if(i==6)
13             break;
14         else
15             cout<<i<<endl;
16     }
17 }
```

```
"C:\Users\Khale\OneDrive\Desktop\ms
10
1
3
4
5
6
7
```

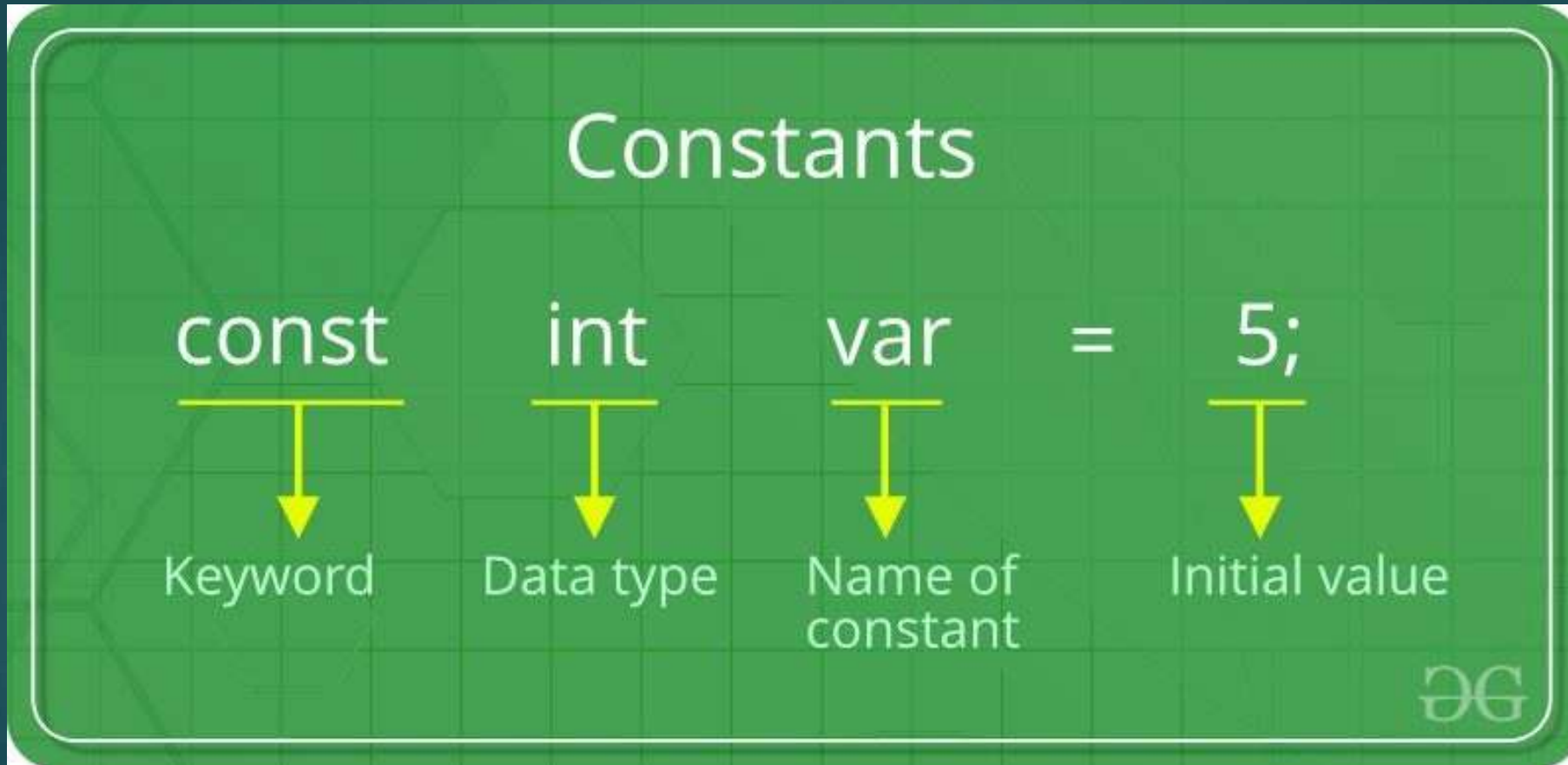
Problems

- ▶ Even Numbers
- ▶ Problem - A

Constant variable

- ▶ What is Constant Variable
- ▶ A Constant variable in C++ is a variable whose value does not change during the execution of the program. Once a value is assigned, in a constant variable, it remains fixed throughout the program.
- ▶ A Constant variable is declared by keyword `const`.
- ▶ Syntax of Declaring Constant Variable in C++
- ▶ `const datatype variable_name;`
- ▶ Here `const` is used for declaring Constant, `datatype` is the name of data type such as `int`, `float`, `double` or `char` and `variable_name` is the name of variable (you can use any name of your choice for example: `a`, `b`, `c`, `alpha`, etc.) and `;` is used for line terminator (end of line)

Constant variable



Global & scope of variable

- ▶ In C++, a global variable is defined as a variable that can be used or accessed from anywhere in the entire program,
- ▶ which is one of the scope types in any programming language.
- ▶ Where the global variable scope is the extent of the program code with in which the variables can be accessed or defined or declared

```
1 #include <iostream>
2 // using namespace std; واحد رخم غايلز يتعب نفسه ويكتب كل شويه
3 float g = 20;
4 int main () {
5     std::cout<<"A simple program for demonstrating global variables in C++"<<endl;
6     std::cout<<"\n"<<endl;
7     float p;
8     p = 10;
9     std::cout << "The value of the local variable p is " << p <<endl;
10    g++;
11    std::cout<<"\n"<<endl;
12    std::cout << "The value of the global variable g is " <<g <<endl;
13    return 0;
14}
```


Functions

Functions

- ▶ **Functions are reusable sections of code that serve a particular purpose.**
- Functions can take inputs and outputs and can be reused across programs.
- Organizing programs into functions helps to organize and simplify code.
- This is an example of abstraction; after you've written a function, you can use the function without having to worry about the details of how the function is implemented. Because of abstraction, others can use (or "call") the function without knowing its lower-level details as well.

Function Declaration Syntax

- ▶ `int raiseToPower(int base, int power)`
- ▶ `{`
 - ▶ `int result = 1;`
 - ▶ `for (int i = 0; i < power; i = i+ 1)`
 - ▶ `{`
 - ▶ `result = result * base;`
 - ▶ `}`
 - ▶ `return result;`
- ▶ `}`
- ▶ Return type-function name-parameters(**these 3 called function signature or function header**)

Functions syntax

- ▶ If the return type anything except void so there should be a return statement of function that return the same value of return type in the header of the function
- ▶ Function signature +body=function declaration
- ▶ raiseToPower(3, 4);
- ▶ The last statement is function invocation or function call

Function Declaration

- ▶ Function declarations need to occur before invocations.

▶ For example:

```
▶ int foo()
```

```
▶ {
```

```
    ▶ return bar()*2; // ERROR; bar hasn't been declared yet
```

```
▶ }
```

```
▶ int bar()
```

```
▶ {
```

```
    ▶ return 3;
```

```
▶ }
```

Function Declaration

- **Function declarations need to occur before invocations.**
- **Solution 1: reorder function declarations.**

▶ For example:

▶ `int bar()`

▶ `{`

▶ `return 3;`

▶ `}`

▶ `int foo()`

▶ `{`

▶ `return bar()*2; // Ok`

Function Declaration

- Function declarations need to occur before invocations.
- Solution 1: reorder function declarations.
- Solution 2: use a function prototype; it informs the compiler that you will implement it later.

▶ For example:

```
▶ int bar();//prototype
▶ int foo()
▶ {
    ▶ return bar()*2; // Ok
▶ }
▶ int bar()
▶ {
    ▶ return 3;
▶ }
```

Function Prototype

- ▶ Function prototypes should match the signature of the method, though argument names don't matter.

```
int square( int );
```

```
int cube( int x )  
{  
    return x*square(x);  
}
```

```
int square( int x )  
{  
    return x*x;  
}
```

```
int square( int x );
```

```
int cube( int x )  
{  
    return x*square(x);  
}
```

```
int square( int x )  
{  
    return x*x;  
}
```

```
int square( int z );
```

```
int cube( int x )  
{  
    return x*square(x);  
}
```

```
int square( int x )  
{  
    return x*x;  
}
```

Returning Values and Arguments' Types

- ▶ Returning a Value

- ▶ **For example:**

- ▶ `int foo()`
 - ▶ `{`
 - ▶ `return "hello"; // error`
 - ▶ `}`

- ▶ **On the other hand:**

- ▶ `int foo()`
 - ▶ `{`
 - ▶ `return 2; // ok`
 - ▶ `}`

Argument Type Matters?

```
▶ void printOnNewLine( int x )  
▶ {  
    ▶ Cout<<x;  
▶ }
```

- *printOnNewLine(3)* works
- *printOnNewLine("hello")* will not compile

Argument Type Matters?

▶ `void printOnNewLine(int x)`

▶ {

`cout<<x;`

▶ }

▶ `void printOnNewLine(char x)`

▶ {

`cout<<x;`

▶ }

- `printOnNewLine(3)` works
- `printOnNewLine('A')` also works

Function Overloading

Many functions with the same name, but different arguments.

The function called is the one whose arguments match the invocation.

For example:

- `void printOnNewLine(int x)`
- `{`
- `cout<<x;`
- `}`
- `void printOnNewLine(char x)`
- `{`
- `cout<<x;`
- `}`
- `printOnNewLine(3)` prints "Integer: 3"
- `printOnNewLine('A')` prints "Character: A"

Function Overloading

Many functions with the same name, but different arguments.

The function called is the one whose arguments match the invocation.

For example:

- `void printOnNewLine(int x)`
- `{`
- `cout<<x;`
- `}`
- `void printOnNewLine(int x, int y)`
- `{`
- `cout<<x<<y;`
- `}`
- `printOnNewLine(3)` prints "One Integer: 3"
- `printOnNewLine(4, 5)` prints "Two Integers: 4,5"

Problems

- ▶ 1) Write a program that finds the sum of the first n natural numbers
- ▶ Input: 10
- ▶ Output: 55

question 2 :teamwork

- ▶ Write a program that will:
 - ▶ • Prompt the user to input ten integer values.
 - ▶ • Calculate the smallest and the greatest of those values.
 - ▶ • Call a function to calculate the difference between those smallest and greatest values.
 - ▶ • Finally: displays the entered ten integers, the difference between the smallest and greatest values,